congruence for natural numbers

Johan G. F. Belinfante and Claudia D. Huang
2005 July 17

```
In[1]:= SetDirectory["i:"]; << goedel71.16a; << tools.m
```

```
:Package Title: goedel71.16a          2005 July 16 at 5:55 p.m.

It is now:  2005 Jul 17 at 7:26

Loading Simplification Rules

TOOLS.M                       Revised 2005 July 13

weightlimit = 40
```

summary

In this notebook rewrite rules about congruence of natural numbers are developed to enable the **GOEDEL** program to automatically recognize the validity of some theorems in Quaife's **Q** and **DV** groups.

```
In[2]:= "Art Quaife, Automated Development of Fundamental Mathematical Theories,
           Kluwer Academic Publishers, Dordrecht, the Netherlands, 1992.";
```

Congruence is a basic concept in number theory, but it is usually developed in the setting of integer arithmetic. In Quaife's work, as well as in this notebook, the subject is taken up in the somewhat more elementary context of natural number arithmetic. On account of this, one needs to be careful with respect to subtraction. In the **GOEDEL** program, the difference of two natural numbers **x** and **y** is denoted by **natsub[x, y]**. When either **x** or **y** fails to be a natural number, or when **y** is greater than **x**, then **natsub[x, y] = V**. The floored difference **monus[x, y]** on the other hand, is zero under these conditions.

```
In[3]:= monus[x, y]
```

```
Out[3]= nat[natsub[x, y]]
```

Both **natsub** and **monus** are used in this notebook, the latter only to compare with Quaife's work. The **nat** wrapper has the property that **nat[x] = x** when **x** is a natural number, and **nat[x] = 0** otherwise. One can view **nat[x]** as a generic natural number.

Using **nat** wrappers allows one to dispense with some numberhood literals, and often simplifies rewrite rules.

---

congruence

If **x** and **y** are natural numbers, then **z = natmod[x,y]** is by definition the least natural number for which the difference **(x – z)** is divisible by **y**. The binary function **NATMOD** takes **pair[x,y]** to **z**, and the composite function **composite[NATMOD, RIGHT[y]]** takes **x** to **z**.

```
In[4]:= member[pair[x, z], composite[NATMOD, RIGHT[y]]]

Out[4]= and[equal[z, natmod[x, y]], member[x, omega], member[y, omega]]
```

We shall say that natural numbers **x** and **y** are **congruent modulo z** if **natmod[x,z]** and **natmod[y,z]** are equal. This definition immediately implies that congruence modulo any given number is an equivalence relation:

```
In[5]:= SubstTest[EQUIVALENCE, composite[inverse[funpart[w]], funpart[w]],
         w → composite[NATMOD, RIGHT[x]]]

Out[5]= EQUIVALENCE[
          composite[inverse[RIGHT[x]], inverse[NATMOD], NATMOD, RIGHT[x]]] == True

In[6]:= EQUIVALENCE[
          composite[inverse[RIGHT[x_]], inverse[NATMOD], NATMOD, RIGHT[x_]]] := True
```

Quaife's theorem (DV22)

This section is devoted to a derivation of Quaife's theorem **(DV22)**, which states that if two natural numbers are congruent modulo **z**, then their floored difference is divisible by **z**. The first lemma uses the fact that if two numbers have a common factor, then their difference also has that factor. This is specialized here to the case that something cancels out in the difference:

```
In[7]:= SubstTest[implies,
         and[member[pair[w, u], DIV], member[pair[w, v], DIV], not[member[u, v]]],
         member[pair[w, natsub[u, v]], DIV],
         {u → natsub[nat[z], natmod[nat[x], nat[y]]],
          v -> natsub[nat[x], natmod[nat[x], nat[y]]], w → nat[y]}]

Out[7]= or[member[natsub[nat[z], natmod[nat[x], nat[y]]],
           natsub[nat[x], natmod[nat[x], nat[y]]]],
          member[pair[nat[y], natsub[nat[z], nat[x]]], DIV], not[
           member[pair[nat[y], natsub[nat[z], natmod[nat[x], nat[y]]]], DIV]]] == True

In[8]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

The transitive law for $\leq$ has this corollary, which is here made into a temporary rewrite rule:

```
In[9]:= equiv[or[member[nat[x], nat[y]], member[nat[z], nat[x]],
          member[nat[z], nat[y]]], or[member[nat[x], nat[y]], member[nat[z], nat[x]]]]

Out[9]= True

In[10]:= or[member[nat[x_], nat[y_]],
           member[nat[z_], nat[x_]], member[nat[z_], nat[y_]]] :=
          or[member[nat[x], nat[y]], member[nat[z], nat[x]]]
```

For natural numbers, the statement that $\mathbf{x} \leq \mathbf{y}$ is true can be replaced with the equivalent statement that $\mathbf{y} < \mathbf{x}$ is false. Specializing to the case that the numbers in question are certain differences yields:

```
In[11]:= SubstTest[implies, and[member[u, omega], member[v, omega], subclass[u, v]],
           not[member[v, u]], {u -> natsub[nat[x], nat[w]],
            v -> natsub[nat[z], nat[w]]}] /. w -> natmod[nat[x], nat[y]]

Out[11]= or[member[nat[z], nat[x]], not[member[natsub[nat[z], natmod[nat[x], nat[y]]],
             natsub[nat[x], natmod[nat[x], nat[y]]]]]] == True

In[12]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Theorem. If natural numbers $\mathbf{x}$ and $\mathbf{y}$ are congruent modulo $\mathbf{z}$, then either $\mathbf{x} - \mathbf{y}$ is divisible by $\mathbf{z}$ or else $\mathbf{x} < \mathbf{y}$.

```
In[13]:= Map[not, SubstTest[and, implies[and[p1, p2], p3], implies[p3, or[p4, p5]],
           implies[p2, not[p4]], not[implies[and[p1, p2], p5]],
           {p1 -> equal[natmod[nat[x], nat[z]], natmod[nat[y], nat[z]]],
            p2 → not[member[nat[x], nat[y]]],
            p3 → member[pair[nat[z], natsub[nat[x], natmod[nat[y], nat[z]]]], DIV],
            p4 -> member[natsub[nat[x], natmod[nat[y], nat[z]]],
              natsub[nat[y], natmod[nat[y], nat[z]]]],
            p5 -> member[pair[nat[z], natsub[nat[x], nat[y]]], DIV]}]]

Out[13]= or[member[nat[x], nat[y]], member[pair[nat[z], natsub[nat[x], nat[y]]], DIV],
           not[equal[natmod[nat[x], nat[z]], natmod[nat[y], nat[z]]]] == True

In[14]:= or[member[nat[x_], nat[y_]],
           member[pair[nat[z_], natsub[nat[x_], nat[y_]]], DIV],
           not[equal[natmod[nat[x_], nat[z_]], natmod[nat[y_], nat[z_]]]]] := True
```

Quaife's **(DV22)** is a consequence. To prove it, we need a lemma relating **natsub** to **monus**.

```
In[15]:= SubstTest[implies,
          and[equal[u, v], member[pair[y, u], DIV]], member[pair[y, v], DIV],
          {u → natsub[nat[z], nat[x]], v -> monus[nat[z], nat[x]]}]

Out[15]= or[member[nat[z], nat[x]], member[pair[y, nat[natsub[nat[z], nat[x]]]], DIV],
           not[member[pair[y, natsub[nat[z], nat[x]]], DIV]] == True

In[16]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

If **monus[x,y]** is zero, then any natural number divides it.

```
In[17]:= SubstTest[implies, equal[w, 0],
          member[pair[nat[z], w], DIV], w → monus[nat[x], nat[y]]]

Out[17]= or[member[nat[y], nat[x]],
           member[pair[nat[z], nat[natsub[nat[x], nat[y]]]], DIV]] == True

In[18]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Quaife's Theorem **(DV22)** follows:

```
In[19]:= Map[not, SubstTest[and, implies[and[p1, p2], p3], implies[and[p2, p3], p5],
           implies[not[p2], not[p4]], implies[not[p4], p5], not[implies[p1, p5]],
           {p1 -> equal[natmod[nat[x], nat[z]], natmod[nat[y], nat[z]]],
            p2 → not[member[nat[x], nat[y]]],
            p3 -> member[pair[nat[z], natsub[nat[x], nat[y]]], DIV],
            p4 → member[nat[y], nat[x]],
            p5 → member[pair[nat[z], monus[nat[x], nat[y]]], DIV]}]]

Out[19]= or[member[pair[nat[z], nat[natsub[nat[x], nat[y]]]], DIV],
           not[equal[natmod[nat[x], nat[z]], natmod[nat[y], nat[z]]]]] == True

In[20]:= or[member[pair[nat[z_], nat[natsub[nat[x_], nat[y_]]]], DIV],
           not[equal[natmod[nat[x_], nat[z_]], natmod[nat[y_], nat[z_]]]]] := True
```

---

a partial converse for (DV22)

The converse of **(DV22)** is of course false: for example, the floored difference **monus[5, 7] = 0** is divisible by **3**, but **5** is not congruent to **7** modulo **3**. In this section a partial converse is derived, using **natsub** instead of **monus**. It is shown that if a difference of two natural numbers **x** and **z** is divisible by **y**, then they are congruent modulo **y**. In this section one can do with fewer **nat** wrappers because the divisibility hypotheses imply that the variables refer to numbers. For example:

```
In[21]:= Map[or[member[x, omega], #] &, SubstTest[implies, member[pair[y, w], DIV],
            member[w, omega], w -> natsub[z, natmod[x, y]]]] // MapNotNot

Out[21]= or[member[x, omega], not[member[pair[y, natsub[z, natmod[x, y]]], DIV]]] == True

In[22]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

The following temporary rewrite rule eliminates a redundant numberhood hypothesis below.

```
In[23]:= equiv[or[not[member[x, omega]],
           not[member[pair[y, natsub[z, natmod[x, y]]], DIV]]],
          not[member[pair[y, natsub[z, natmod[x, y]]], DIV]]]

Out[23]= True

In[24]:= or[not[member[x_, omega]],
           not[member[pair[y_, natsub[z_, natmod[x_, y_]]], DIV]]] :=
          not[member[pair[y, natsub[z, natmod[x, y]]], DIV]]
```

Application of the uniqueness theorem for **natmod**.

```
In[25]:= SubstTest[implies, and[member[pair[v, natsub[u, w]], DIV], member[w, v]],
          equal[w, natmod[u, v]], {u → z, v → y, w → natmod[x, y]}]

Out[25]= or[equal[0, y], equal[natmod[x, y], natmod[z, y]],
           not[member[pair[y, natsub[z, natmod[x, y]]], DIV]]] == True

In[26]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

The hypothesis that **y** is not zero is redundant, and can be eliminated.

```
In[27]:= SubstTest[and, implies[and[p1, p2], p3], implies[p1, or[p2, p3]],
           {p1 -> member[pair[y, natsub[z, natmod[x, y]]], DIV],
            p2 → equal[0, y], p3 -> equal[natmod[x, y], natmod[z, y]]}] // Reverse

Out[27]= or[equal[natmod[x, y], natmod[z, y]],
           not[member[pair[y, natsub[z, natmod[x, y]]], DIV]]] == True

In[28]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

One final lemma.

```
In[29]:= Map[or[#, member[pair[y, natsub[z, natmod[x, y]]], DIV]] &,
          SubstTest[implies, and[member[pair[y, u], DIV], member[pair[y, v], DIV]],
           member[pair[y, natadd[u, v]], DIV],
           {u → natsub[z, x], v → natsub[x, natmod[x, y]]}]]

Out[29]= or[member[pair[y, natsub[z, natmod[x, y]]], DIV],
           not[member[pair[y, natsub[z, x]], DIV]]] == True

In[30]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

The main result resembles a converse of **(DV22)** but note that it uses **natsub** instead of **monus**.

```
In[31]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
           not[implies[p1, p3]], {p1 -> member[pair[y, natsub[z, x]], DIV],
            p2 -> member[pair[y, natsub[z, natmod[x, y]]], DIV],
            p3 -> equal[natmod[x, y], natmod[z, y]]}]]

Out[31]= or[equal[natmod[x, y], natmod[z, y]],
           not[member[pair[y, natsub[z, x]], DIV]]] == True

In[32]:= or[equal[natmod[x_, y_], natmod[z_, y_]],
           not[member[pair[y_, natsub[z_, x_]], DIV]]] := True
```

corollary needed for (Q2) and (Q4)

The theorem proved in the preceding section implies the following:

```
In[33]:= SubstTest[implies, member[pair[y, natsub[u, v]], DIV],
          equal[natmod[u, y], natmod[v, y]], {u -> natadd[natmul[nat[x], y], z], v → z}]

Out[33]= or[equal[natmod[z, y], natmod[natadd[z, natmul[y, nat[x]]], y]],
           not[member[y, omega]], not[member[z, omega]]] == True

In[34]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

The numberhood literals are redundant, and can be removed as follows:

```
In[35]:= SubstTest[implies, and[equal[u, V], equal[V, v]], equal[u, v],
          {u -> natmod[z, y], v -> natmod[natadd[z, natmul[nat[x], y]], y]}]

Out[35]= or[and[member[y, omega], member[z, omega]],
           equal[natmod[z, y], natmod[natadd[z, natmul[y, nat[x]]], y]]] == True

In[36]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Main result.

```
In[37]:= SubstTest[and, implies[p1, p2], implies[not[p1], p2],
          {p1 -> and[member[y, omega], member[z, omega]],
           p2 -> equal[natmod[z, y], natmod[natadd[z, natmul[y, nat[x]]], y]]}]

Out[37]= True == equal[natmod[z, y], natmod[natadd[z, natmul[y, nat[x]]], y]]

In[38]:= natmod[natadd[z_, natmul[y_, nat[x_]]], y_] := natmod[z, y]
```

Further corollary. (Case of two factors, needed for **(Q9)**.)

```
In[39]:= SubstTest[natmod, natadd[z, natmul[y, nat[w]]], y, w → natmul[nat[u], nat[v]]]

Out[39]= natmod[natadd[z, natmul[y, nat[u], nat[v]]], y] == natmod[z, y]

In[40]:= natmod[natadd[z_, natmul[y_, nat[u_], nat[v_]]], y_] := natmod[z, y]
```

This rewrite rule suffices for the **GOEDEL** program to recognize the validity of the second clauses of Quaife's **(Q2)** and the first demodulator of Quaife's **(Q4)**. (Not all variables need explicit **nat** wrappers.)

In[41]:=
```
implies[and[equal[natadd[v, natmul[nat[y], nat[u]]], x], member[v, nat[y]]],
 equal[natmod[x, nat[y]], v]]
```

Out[41]= True

In[42]:=
```
Equal[natmod[natadd[natmod[v, y], natmul[y, nat[u]]], y], natmod[v, y]]
```

Out[42]= True